ECCOMAS Congress 2016
VII European Congress on Computational Methods in Applied Sciences and Engineering
M. Papadrakakis, V. Papadopoulos, G. Stefanou, V. Plevris (eds.)
Crete Island, Greece, 5–10 June 2016

PRECONDITIONERS FOR ISOGEOMETRIC ANALYSIS BASED ON SOLVERS FOR THE SYLVESTER EQUATION

Giancarlo Sangalli[1] **and Mattia Tani**[1]

[1] Università di Pavia, Dipartimento di Matematica "F. Casorati"
Via A. Ferrata 1, 27100 Pavia, Italy
e-mail: {sangia05,mattia.tani}@unipv.it

Keywords: Isogeometric analysis, preconditioning, Kronecker product, Sylvester equation.

Abstract. *In this work, which represents a condensed version of [1], we consider large linear systems arising from the isogeometric discretization of the Poisson problem on a single-patch domain. We consider a preconditioning strategy which is based on the solution of a Sylvester-like equation at each step of an iterative solver. This strategy, which fully exploits the tensor structure that underlies isogeometric problems, is robust with respect to both mesh size and spline degree. The application of the preconditioner is performed by a popular direct solver for the Sylvester equation, whose implementation details are given in the 2D and 3D case, with particular emphasis on the latter. We show numerical experiments for 3D problems, which demonstrate the potential of this approach.*

1 Introduction

The present manuscript represents a condensed version of [1]. We are concerned with the problem of solving large linear systems arising from isogeometric discretizations [2]. An extension of the classical finite element method, the isogeometric method based on the idea of using splines or other functions constructed from splines (e.g., non-uniform rational B-splines, NURBS) both for the parametrization of the computational domain, as it is typically done by computer aided design software, and for the representation of the unknown solution fields of the PDE of interest. Unlike standard finite element methods, the isogeometric method makes possible to use high-regularity functions. The so-called isogeometric k-method, based on splines of degree p and global C^{p-1} regularity, has shown significant advantages in term of higher accuracy per degree-of-freedom in comparison to C^0 finite elements of degree p [3, 4].

The study of the computational efficiency of linear solvers for isogeometric discretizations has been initiated in the papers [5] and [6], where it has been shown that the algorithms used with the finite element method suffer of performance degradation when used used in the context of the isogeometric k-method. Since then, a number of papers have been published that analyse and develop preconditioners for isogeometric linear systems, see e.g. [7, 8, 9, 10]. While containing important advances, these papers also confirms the difficulty in achieving both *robustness* and computational *efficiency* for the high-degree k-method.

Robust multigrid preconditioners have been proposed in the recent papers [11] and [12]. We emphasize that, in both works, the authors exploit the tensor-product structure of isogeometric spaces in order to achieve efficiency in multidimensional problems.

In this paper we also exploit the tensor-product structure of multivariate spline space, on a different basis. We consider the Bartels-Stewart direct solver, a method that has been developed for the so-called Sylvester equation, and discuss its application as a preconditioner for isogeometric systems. We describe this method in both 2D and 3D cases. We then focus on 3D problems, discussing its computational cost and showing numerical experiments which assess the potential of this approach.

In 3D, application of the Bartels-Stewart solver requires $O(N^{4/3})$ FLOPs. However, in all our benchmarks that uses a conjugate gradient (CG) iterative solver, the computational time spent in the Bartels-Stewart preconditioner application is even lower than the residual computation (multiplication matrix $\mathcal{A}$ times a vector). This surprising performance is due to the fact that the Bartels-Stewart solver requires dense matrix-matrix operations that takes advantages of modern computer architecture. Furthermore, the Bartels-Stewart method is especially suited to parallelisation which may significantly speed up the execution time, though this is not considered in our analysis.

We emphasize that the proposed approach is robust, in the sense that the condition number of the preconditioned system is uniformly bounded with respect to the degree p and mesh size h when the parametrization is regular. In all cases, it is important to have strategies to further improve the condition number, and this will be the topic of further researches.

2 The problem

We consider, as a model problem, the Poisson equation with Dirichlet boundary conditions:

$$\begin{cases} -\mathrm{div}(K(\mathbf{x})\nabla \mathrm{u}(\mathbf{x})) = f(\mathbf{x}) \quad \text{on} \quad \Omega \subseteq \mathbb{R}^d \\ \qquad\qquad\qquad\qquad \mathrm{u} = 0 \quad \text{on} \quad \partial\Omega \end{cases} \tag{1}$$

where $K(\mathbf{x})$ is a symmetric positive definite matrix for each $\mathbf{x} \in \Omega$. In isogeometric methods, Ω is given by a spline or NURBS parametrization. For the sake of simplicity, we consider a single-patch spline parametrization $\boldsymbol{F}$ which satisfies $\Omega = \boldsymbol{F}\left([0,1]^d\right)$.

Following the isogeometric approach, we consider the Galerkin discretization of (1) relative to a space of multivariate B-splines on Ω. We recall that multivariate B-splines in dimension d ($d = 2, 3$ are the interesting cases) are defined from univariate B-splines by tensorization. We assume for simplicity that the spline degree p and the number n of univariate basis functions which vanish at the boundary is the same in all directions. We are primarily interested in the so called k-refinement or isogeometric k-method [2], hence we consider C^{p-1} continuous splines, i.e. splines with maximum regularity.

Let $N = n^3$ and let $B_1, \dots, B_N$, denote the basis function of the isogeometric space, which incorporate the homogeneous Dirichlet boundary conditions. Then the Galerkin stiffness matrix reads

$$\begin{aligned} \mathcal{A}_{ij} &= \int_\Omega \left(\nabla B_i(\boldsymbol{x})\right)^T K(\boldsymbol{x}) \nabla B_j(\boldsymbol{x}) d\boldsymbol{x} \\ &= \int_{[0,1]^d} \left(\nabla \hat{B}_i(\boldsymbol{\xi})\right)^T Q(\boldsymbol{\xi})\, \nabla \hat{B}_j\left(\boldsymbol{\xi}\right)\, d\boldsymbol{\xi}, \qquad i,j = 1, \dots, N \end{aligned} \tag{2}$$

where

$$Q = \det\left(J_{\boldsymbol{F}}\right) J_{\boldsymbol{F}}^{-T} K J_{\boldsymbol{F}}^{-1} \tag{3}$$

and $J_{\boldsymbol{F}}$ denotes the Jacobian of $\boldsymbol{F}$.

The support of each basis spline that does not touch $\partial\Omega$ intersects the support of $(2p+1)^d$ basis splines (including itself), while the support of a basis spline intersecting $\partial\Omega$ overlaps at least $(p+1)^d$ and up to $(2p+1)^d$ basis splines supports (including itself). Thus, the number of nonzeros of $\mathcal{A}$ is about $(2p+1)^d N$.

3 The preconditioner

Consider the matrix

$$\mathcal{P}_{ij} = \int_{[0,1]^d} \left(\nabla \hat{B}_i\right)^T \nabla \hat{B}_j \, d\xi, \qquad i,j = 1, \dots, N \tag{4}$$

Observe that $\mathcal{P} = \mathcal{A}$ in the special case when K is the identity matrix and $\boldsymbol{F}$ is the identity function, which means that $\Omega = [0,1]^d$.

By exploiting the tensor product structure of the basis functions, the matrix $\mathcal{P}$ can be expressed as the sum of d Kronecker products. For example, when $d = 2$ we have

$$\mathcal{P} = K_1 \otimes M_2 + M_1 \otimes K_2$$

where M_1, M_2 represent the mass, and K_1, K_2 the stiffness univariate matrices. Such matrices are all symmetric positive definite and banded with bandwidth p (we say that a matrix B has bandwidth p if $B_{ij} = 0$ for $|i-j| > p$). These matrices have the same order n. Similarly, when $d = 3$

$$\mathcal{P} = K_1 \otimes M_2 \otimes M_3 + M_1 \otimes K_2 \otimes M_3 + M_1 \otimes M_2 \otimes K_3.$$

By comparing (2) and (4), observe that $\mathcal{P}_{ij} \neq 0$ if and only if $\mathcal{A}_{ij} \neq 0$. Thus, despite having different entries in general, $\mathcal{A}$ and $\mathcal{P}$ have the same sparsity pattern.

We propose $\mathcal{P}$, defined in (4), as a preconditioner for the isogeometric matrix $\mathcal{A}$. In other words, we want to precondition a problem with arbitrary geometry and coefficients with a solver for the same operator on the parameter domain, with constant coefficients. This is a common approach, see e.g. [13], [11] and [12].

It is known (see e.g. [1] for a proof) that

$$\kappa\left(\mathcal{P}^{-1}\mathcal{A}\right) \leq \frac{\sup_{\Omega} \lambda_{\max}\left(Q\right)}{\inf_{\Omega} \lambda_{\min}\left(Q\right)} \tag{5}$$

where the matrix Q is given in (3).

As long as the considered problem doesn't depart much from the model problem on the square with constant coefficients, the right-hand side of (5) will be small and the preconditioner is expected to perform well. On the other hand, if the eigenvalues of Q vary widely, due to the presence of complicated geometry or coefficients, the preconditioner performance decreases. In these cases, it is useful to have strategies to improve the spectral conditioning of $\mathcal{P}^{-1}\mathcal{A}$: this is a topic that we will address in a forthcoming paper. We emphasize that bound (5) does not depend neither on the mesh size nor on the spline degree, but only on $\boldsymbol{F}$ and K.

4 The Bartels-Stewart method

If the system $\mathcal{A}u = b$ is solved using the Preconditioned Conjugate Gradient (PCG) method, at each iteration we need to solve a system of the form

$$\mathcal{P}s = r \tag{6}$$

where r is the current residual. The computational effort required for solving this system (exactly or approximately) is of paramount importance in assessing the overall performance of PCG.

When $d = 2$, equation (6) can be reformulated as

$$K_1 S M_2 + M_1 S K_2 = R \tag{7}$$

where $r = \text{vec}(R)$ and $s = \text{vec}(S)$, with the vec operator stacking the rows of a matrix $X \in \mathbb{R}^{n\times n}$ into a single column vector $x \in \mathbb{R}^{n^2}$. Equation (7) takes the name of (generalized) Sylvester equation. Due to its many applications, the literature dealing with Sylvester equation (and its variants) is vast, and a number of methods have been proposed for its numerical solution. We refer to [14] for a recent survey on this subject.

In this paper we consider the Bartels-Stewart method [15], which represents the state-of-the-art direct solver for (7). In fact, the method considered here departs slightly from the standard Bartels-Stewart solvers in that it exploits the symmetry of the coefficient matrices M_1, M_2, K_1, K_2, and was first presented in [16]. However, due to its popularity in the context of the Sylvester equation, we still refer to this approach as the Bartels-Stewart method. We remark that in [1] an iterative approach, namely the Alternating Direction Implicit (ADI) method is also considered. However, it yields a worst performance, especially for 3D problems, than the direct method, and hence we do not discuss it here.

We first describe the Bartels-Stewart method in the 2D case, where the system (6) reads

$$\left(K_1 \otimes M_2 + M_1 \otimes K_2\right) s = r \tag{8}$$

We consider the generalized eigendecomposition of the matrix pencils (K_1, M_1) and (K_2, M_2), namely

$$K_1 U_1 = M_1 U_1 D_1 \qquad K_2 U_2 = M_2 U_2 D_2 \tag{9}$$

where D_1 and D_2 are diagonal matrices whose entries are the eigenvalues of $M_1^{-1} K_1$ and $M_2^{-1} K_2$, respectively, while U_1 and U_2 satisfy

$$U_1^T M_1 U_1 = I, \qquad U_2^T M_2 U_2 = I,$$

which implies in particular $U_1^{-T} U_1^{-1} = M_1$ and $U_2^{-T} U_2^{-1} = M_2$, and also, from (9), $U_1^{-T} D_1 U_1^{-1} = K_1$ and $U_2^{-T} D_2 U_2^{-1} = K_2$. Therefore we factorize $\mathcal{P}$ in (8) as follows:

$$(U_1 \otimes U_2)^{-T} (D_1 \otimes I + I \otimes D_2) (U_1 \otimes U_2)^{-1} s = r,$$

and adopt the following strategy:

Algorithm 1 Bartels-Stewart direct method (symmetric case)

1: Compute the generalized eigendecompositions (9)
2: Compute $\tilde{r} = (U_1 \otimes U_2)^T r$
3: Compute $\tilde{s} = (D_1 \otimes I + I \otimes D_2)^{-1} \tilde{r}$
4: Compute $s = (U_1 \otimes U_2) \tilde{s}$

This approach can be generalized to the 3D case, where (6) takes the form

$$(K_1 \otimes M_2 \otimes M_3 + M_1 \otimes K_2 \otimes M_3 + M_1 \otimes M_2 \otimes K_3)\, s = r \tag{10}$$

in a straightforward way. We consider the generalized eigendecompositions

$$K_1 U_1 = M_1 U_1 D_1, \qquad K_2 U_2 = M_2 U_2 D_2, \qquad K_3 U_3 = M_3 U_3 D_3 \tag{11}$$

with D_1, D_2, D_3 diagonal matrices and

$$U_1^T M_1 U_1 = I, \qquad U_2^T M_2 U_2 = I, \qquad U_3^T M_3 U_3 = I.$$

Then, (10) can be factorized as

$$(U_1 \otimes U_2 \otimes U_3)^{-1} (D_1 \otimes I \otimes I + I \otimes D_2 \otimes I + I \otimes I \otimes D_3) (U_1 \otimes U_2 \otimes U_3)^{-T} s = r,$$

which suggests the following algorithm.

Algorithm 2 3D Bartels-Stewart method (symmetric case)

1: Compute the generalized eigendecompositions (11)
2: Compute $\tilde{r} = (U_1 \otimes U_2 \otimes U_3) r$
3: Compute $\tilde{s} = (D_1 \otimes I \otimes I + I \otimes D_2 \otimes I + I \otimes I \otimes D_3)^{-1} \tilde{r}$
4: Compute $s = (U_1 \otimes U_2 \otimes U_3)^T \tilde{s}$

4.1 Computational cost for the 3D approach

We discuss in detail the computational cost of the Bartels-Stewart method only in the 3D case, i.e. Algorithm 2, as this represents the focus of the present manuscript. The exact cost of the eigendecompositions in line 1 depends on the algorithm employed, however it is safe to assume that this step costs $O(n^3)$ FLOPs. We further mention that when a divide-and-conquer approach is used [17, Chapter 8], the eigendecomposition can be naturally performed in parallel computer architectures. Moreover, when Algorithm 2 is applied as a preconditioner for CG, then this step may be performed only once, since the matrices involved do not change throughout the CG iteration. Line 3 is just a diagonal scaling, and it also requires $O(n^3)$ FLOPs. Lines 2 and 4, as it can be seen by exploiting the properties of Kronecker products [18], are equivalent to performing a total of 6 products between dense matrices of size $n \times n$ and $n \times n^2$. Thus, neglecting lower order terms the overall computational cost of Algorithm 2 is $12n^4$ FLOPs. We emphasize that this cost is independent of p.

We point out that the main computational effort of the method consists in a few (dense) matrix-matrix products, which are level 3 BLAS operations and typically yield high efficiency thanks to a dedicated implementation on modern computers by optimised usage of the memory cache hierarchy [17, Chapter 1]. Matrix-matrix products are also naturally suited for parallelisation. Second, in a preconditioned CG iteration the cost for applying the preconditioner has to be compared with the cost of the residual computation (a matrix-vector product with $\mathcal{A}$) which can be quantified in approximately $2(2p+1)^3n^3$ for 3D problems, resulting in a FLOPs ratio of the preconditioner application to residual computation (matrix-vector product) of $(3n)/(4p^3) \approx n/p^3$. For example, if $N = 256^3$ and $p = 4$, the preconditioner requires only 3 times more FLOPs than the residual computation, while for degree $p = 6$ the matrix-vector product is even more costly than the preconditioner itself. However in numerical tests we will see that, for all cases of practical interest in 3D, the computational time used by the preconditioner application is far lower that the residual computation itself. This is because the computational time depends not only on the FLOPs count but also on the memory usage and, as mentioned above, dense matrix-matrix multiplications greatly benefit of modern computer architecture.

5 3D numerical experiments

We now numerically show the potential of the Bartels-Stewart method in solving isogeometric systems. We only show 3D experiments here, and refer to [1] for experiments in the 2D case.

Algorithm 2 was implemented in MATLAB Version 8.5.0.197613 (R2015a), with the toolbox GEOPDES [19], on a 12x Intel Xeon i7-5820K, 3.30GHz, 64 GB of RAM. We used the MATLAB function `eig` to compute the generalized eigendecomposition (11), and the products involving Kronecker matrices (lines 2 and 4 of Algorithm 2) were performed using the function from the free MATLAB toolbox Tensorlab [20]. We recall MATLAB allows implicit parallelism for some of its operations, e.g. dense matrix multiplication and `eig`.

We first consider a domain with trivial geometry, namely the unit cube $[0,1]^3$, and then turn to more complicated domains. The first is a thick quarter of ring (Figure 1); note this solid has a trivial geometry on the third direction. The second complicated domain is the solid of revolution obtained by a 2D quarter of ring (Figure 2). Specifically, we performed a $\pi/2$ revolution around the axis having direction $(0,1,0)$ and passing through $(-1,-1,-1)$. We emphasize that here the geometry is nontrivial along all directions. In the cube case, we set $b =$`randn`$(n^3,1)$

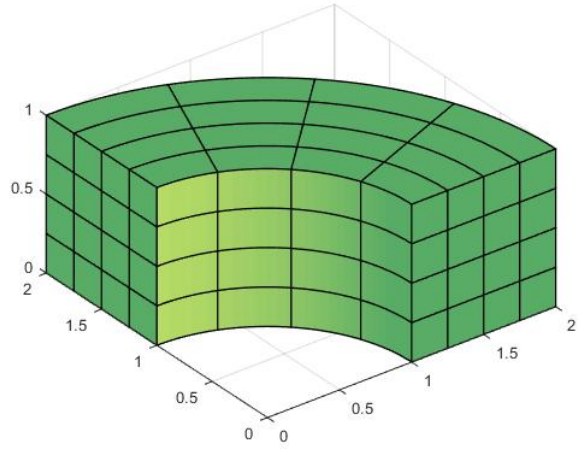

Figure 1: Thick ring domain

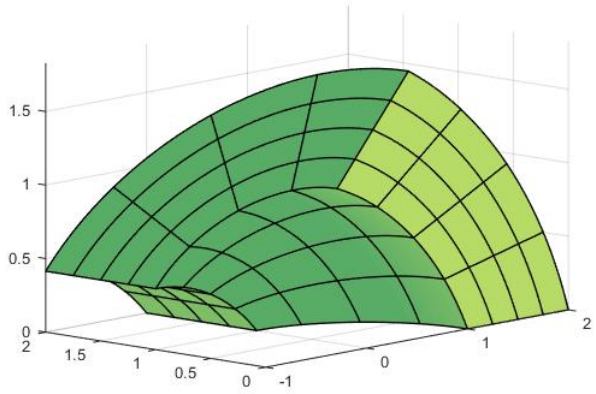

Figure 2: Revolved ring domain

for computational ease, while in the other two cases b is the vector representing the function $f(x, y, z) = 2(x^2 - x) + 2(y^2 - y) + 2(z^2 - z)$. In all problems we set K as the identity matrix, since according to (5) the presence of coefficients and of a nontrivial geometry have an analogous impact on the difficulty of the problem.

We start by considering the problem on the cube domain, which is simply $[0, 1]^3$. As already said, in this case $\mathcal{P} = \mathcal{A}$ and we can directly apply the considered method to the system $\mathcal{A}u = b$. This is not a realistic case but serves as a preliminary check on the proposed theory and implementation. Results for different values of h and p are shown in Table 1. We can see that the computational time required by the direct solver is independent of the degree p. In fact, the timings look impressive and show the great efficiency of this approach. We emphasize that, on the finer discretization level, problems of more than one billion variables is solved in slightly more than one minute (regardless of p).

We now consider the problems with nontrivial geometries, where the two methods are used

	3D Direct Solver Time (seconds)					
h^{-1}	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$	$p=6$
128	0.07	0.06	0.06	0.07	0.08	0.08
256	0.53	0.61	0.64	0.65	0.65	0.56
512	5.58	8.01	6.80	5.83	8.00	5.76
1024	66.79	68.46	67.30	66.38	66.60	63.86

Table 1: Performance of the Bartels-Stewart method, cube domain.

	CG + $\mathcal{P}$ Iterations / Time (seconds)				
h^{-1}	$p=2$	$p=3$	$p=4$	$p=5$	$p=6$
32	26 / 0.21	26 / 0.42	26 / 0.83	26 / 1.63	26 / 2.84
64	27 / 1.46	27 / 3.36	27 / 7.41	27 / 13.44	27 / 22.95
128	28 / 12.74	28 / 30.30	28 / 60.14	*	*

	CG + IC Iterations / Time (seconds)				
h^{-1}	$p=2$	$p=3$	$p=4$	$p=5$	$p=6$
32	21 / 0.36	15 / 1.22	12 / 3.60	10 / 10.22	9 / 27.30
64	37 / 4.65	28 / 13.63	22 / 36.48	18 / 94.42	16 / 242.22
128	73 / 65.91	51 / 168.31	41 / 386.84	*	*

Table 2: Thick quarter of ring domain. Performance of CG preconditioned by the Bartels-Stewart method (upper table) and by Incomplete Cholesky (lower table).

as preconditioners for CG. In both cases, we set 10^{-8} as tolerance on the relative residual for CG. To better judge the efficiency of the Bartels-Stewart approach, we compare the results with those obtained when using a preconditioner based on the Incomplete Cholesky (IC) factorization (implemented by the MATLAB function `ichol`). We remark that incomplete factorizations have been considered as preconditioners for IGA problems in [6], where the authors observed that this approach is quite robust w.r.t. p.

In Table 2 we report the results for the thick quarter ring while in Table 3 we report the results for the revolved ring. The symbol "*" denotes the cases in which even assembling the system matrix $\mathcal{A}$ was unfeasible due to memory limitations. We emphasize that the reported computation time always includes the time needed to setup the preconditioner. Below we report some comments on the numerical results for these two problems.

	CG + $\mathcal{P}$ Iterations / Time (seconds)				
h^{-1}	$p=2$	$p=3$	$p=4$	$p=5$	$p=6$
32	40 / 0.30	41 / 0.62	41 / 1.25	42 / 2.61	42 / 4.47
64	44 / 2.16	44 / 5.21	45 / 11.15	45 / 21.83	45 / 36.60
128	47 / 21.99	47 / 56.58	47 / 113.42	*	*

	CG + IC Iterations / Time (seconds)				
h^{-1}	$p=2$	$p=3$	$p=4$	$p=5$	$p=6$
32	24 / 0.40	18 / 1.31	15 / 3.76	12 / 10.36	11 / 27.01
64	47 / 5.53	35 / 15.70	28 / 40.60	24 / 99.76	20 / 249.54
128	94 / 108.53	71 / 241.43	57 / 889.05	*	*

Table 3: Revolved quarter of ring domain. Performance of CG preconditioned by the Bartels-Stewart method (upper table) and by Incomplete Cholesky (lower table).

h^{-1}	$p=2$	$p=3$	$p=4$	$p=5$	$p=6$
32	24.43	12.40	6.52	4.15	2.66
64	19.60	9.10	3.78	2.21	1.24
128	15.58	7.00	3.73	*	*

Table 4: Percentage of time spent in the application of the 3D Bartels-Stewart preconditioner with respect to the overall CG time. Revolved ring domain.

- For the Bartels-Stewart preconditioner, the number of CG iterations is practically independent on p and slightly increases as the mesh is refined, but stays uniformly bounded according to (5).
- Interestingly, in the IC approach the number of CG iterations decreases for higher p. On the other hand, the CPU time still increases due to the greater computational cost of forming and applying the preconditioner. In particular, the performance of the IC approach shows a stronger dependence on p than the Bartels-Stewart approach.
- The Bartels-Stewart preconditioner yields better performance, in terms of CPU time, than the IC preconditioner for all considered values of h and p.

It is fundamental to observe that the computation times reported in Tables 2 and 3 include the time spent in the residual computation of the outer CG iteration (a sparse matrix-vector product, costing $O(p^3n^3)$ FLOPs). Somewhat surprisingly, in the case of the Bartels-Stewart preconditioner this step actually represents a significant effort in the overall CG performance. In fact, our numerical experience shows that the 3D direct method is so efficient that its cost is often negligible w.r.t. to the cost of the residual computation. This effect is clearly shown in Table 4, where we we report the percentage of time spent in the application of the preconditioner when compared with the overall time of CG, in the case of the revolved ring domain.

6 Conclusions

In this work we have analysed and tested the use of a fast solver for Sylvester-like equation, namely the Bartels-Stewart method, as preconditioner for isogeometric discretizations. We considered here a Poisson problem on a single-patch domain, and we focused on the k-method, i.e., splines with maximal smoothness. The considered preconditioner $\mathcal{P}$ is robust w.r.t. h and p, and we have compared two popular methods for its application. We found that the considered approach is very effective and easily outperform a simple-minded incomplete Cholesky preconditioner.

Our conclusion is that the Bartels-Stewart method represents a very promising preconditioning approach in an iterative solver for isogeometric discretizations. In a forthcoming paper we will further study the role of the geometry parametrization on the performance of the approach based on Sylvester equation solvers, and propose possible strategies to improve it.

REFERENCES

[1] G. Sangalli, and M. Tani, Isogeometric preconditioners based on fast solvers for the Sylvester equation, arXiv preprint arXiv:1602.01636, 2016.

[2] J. A. Cottrell, T. J. R. Hughes, Y. Bazilevs, *Isogeometric analysis: toward integration of CAD and FEA*, John Wiley & Sons, 2009

[3] J. A. Cottrell, T. J. R. Hughes, A. Reali, Studies of refinement and continuity in isogeometric structural analysis, *Computer methods in applied mechanics and engineering*, **196**, 4160–4183, 2007.

[4] L. Beirão da Veiga, A. Buffa, G. Sangalli, R. Vázquez, Mathematical analysis of variational isogeometric methods, *Acta Numerica*, **23**, 157–287, 2014.

[5] N. Collier, D. Pardo, L. Dalcin, M. Paszynski, V. M. Calo, The cost of continuity: a study of the performance of isogeometric finite elements using direct solvers, *Computer Methods in Applied Mechanics and Engineering*, **213**, 353–361, 2012.

[6] N. Collier, L. Dalcin, D. Pardo, V. M. Calo, The cost of continuity: performance of iterative solvers on isogeometric finite elements, *SIAM Journal on Scientific Computing*, **35**, A767–A784, 2013.

[7] A. Buffa, and H. Harbrecht, and A. Kunoth, and G. Sangalli, *BPX-preconditioning for isogeometric analysis*, Computer Methods in Applied Mechanics and Engineering, **265**, 63–70, 2013.

[8] K. P. S. Gahalaut, J. K. Kraus, S. K. Tomar, Multigrid methods for isogeometric discretization, *Computer methods in applied mechanics and engineering*, **253**, 413–425, 2013.

[9] L. Beirão da Veiga, D. Cho, L. F. Pavarino, S. Scacchi, Overlapping Schwarz methods for isogeometric analysis, *SIAM Journal on Numerical Analysis*, **50**, 1394–1416, 2012.

[10] L. Beirão da Veiga, D. Cho, L. F. Pavarino, S. Scacchi, BDDC preconditioners for isogeometric analysis, *Mathematical Models and Methods in Applied Sciences*, **23**, 1099–1142, 2013.

[11] M. Donatelli, and C. Garoni, and C. Manni, and S. Serra-Capizzano, and H. Speleers, Robust and optimal multi-iterative techniques for IgA Galerkin linear systems, *Computer Methods in Applied Mechanics and Engineering*, **284**, 230–264, 2015.

[12] C. Hofreither, and S. Takacs, and W. Zulehner, A Robust Multigrid Method for Isogeometric Analysis using Boundary Correction, *NFN Technical Report 33*, 2015.

[13] L. Gao, Kronecker Products on Preconditioning, PhD Thesis, King Abdullah University of Science and Technology, 2013.

[14] V. Simoncini, Computational methods for linear matrix equations, to appear on *SIAM Review*, 2013.

[15] R. H. Bartels, G. W. Stewart, Solution of the matrix equation AX+ XB= C, *Communications of the ACM*, **15**, 820–826, 1972.

[16] R. E. Lynch, J. R. Rice, T. H. Donald, Direct solution of partial difference equations by tensor product methods, *Numerische Mathematik*, **6**, 185–199, 1964.

[17] G. H. Golub, C. F. Van Loan, *Matrix computations*, JHU Press, 2012.

[18] C. F. Van Loan, The ubiquitous Kronecker product, *Journal of computational and applied mathematics*, **123**, 85–100, 2000.

[19] C. De Falco, A. Reali, and R. Vázquez, GeoPDEs: a research tool for isogeometric analysis of PDEs, *Advances in Engineering Software*, **42**, 1020–1034, 2011.

[20] L. Sorber, M. Van Barel, L. De Lathauwer, Tensorlab v2. 0, Available online, URL: www.tensorlab.net, 2014.